PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G07C 9/00

A1

(11) International Publication Number: WO 96/38820

(43) International Publication Date: 5 December 1996 (05.12.96)

(21) International Application Number: PCT/GB96/01249

(22) International Filing Date: 24 May 1996 (24.05.96)

(30) Priority Data:
9511140.7 2 June 1995 (02.06.95) GB

(71) Applicant (*for all designated States except US*): MAYOR LIMITED [GB/GB]; Optimus, Bellbrook Business Park, Uckfield, East Sussex TN22 1QU (GB).

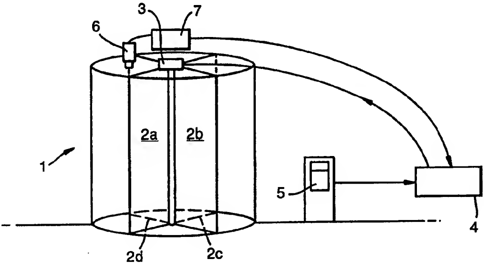
(72) Inventors; and
(75) Inventors/Applicants (*for US only*): FAIRHURST, Michael, Christopher [GB/GB]; 53 The Crescent, Canterbury, Kent CT2 7AW (GB). KELLY, Stephen William [GB/GB]; 51 The Crescent, Canterbury, Kent CT2 7AW (GB). GOLDING, Martin [GB/GB]; Flat 1, 4 Dent-de-Lions, Westgate, Kent CT8 8NT (GB).

(74) Agent: GILL JENNINGS & EVERY; Broadgate House, 7 Eldon Street, London EC2M 7LH (GB).

(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SECURITY CONTROL SYSTEM



(57) Abstract

Control system for a security access device, in particular for revolving doors (1), including video imaging and processing to determine the number of persons within a secured region of the security access device, and controlling the operability of the security access device in dependence on the number of persons.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgystan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SG	Singapore
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LR	Liberia	SZ	Swaziland
CS	Czechoslovakia	LT	Lithuania	TD	Chad
CZ	Czech Republic	LU	Luxembourg	TG	Togo
DE	Germany	LV	Latvia	TJ	Tajikistan
DK	Denmark	MC	Monaco	TT	Trinidad and Tobago
EE	Estonia	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	UG	Uganda
FI	Finland	ML	Mali	US	United States of America
FR	France	MN	Mongolia	UZ	Uzbekistan
GA	Gabon	MR	Mauritania	VN	Viet Nam

SECURITY CONTROL SYSTEM

TECHNICAL FIELD

The present invention relates to a control system for use with a security door, with interlocking or synchronised doors, or with other means, such as turnstiles, flaps or other obstacles, for controlling access to a secured area.

A typical security door may comprise a revolving door divided into, say, four compartments by radially extending wings. The wings are coupled centrally at their upper or lower end to an interlock operated by a control system and are typically motor driven, but may alternatively be pushed manually. Turnstile systems are generally free to be pushed manually.

The control system may operate, for example, in response to a card reader. An authorised person wishing to pass through the door will then insert their pass card in the reader and, provided that their card is recognised, the control system then operates the interlock to free the revolving door so the user can pass through it. If the card is not recognised, or if an unauthorised person attempts to gain access without use of the pass card reader, then the interlock holds the wings of the revolving door against movement and so prevents passage through the door.

BACKGROUND ART

Known security doors suffer from a number of potential forms of misuse. In particular they are vulnerable to "piggy-backing" in which two individuals attempt to pass through the door in one compartment, or "tail-gating" in which an unauthorised person enters the compartment immediately following the one containing the authorised person, or passes through the door in the opposite direction. It has previously been proposed to use pressure-sensitive door mats in the security door, or to use ultra-sonic sensors to detect the presence of more than one person in the door. However these measures have not been wholly successful and there remains a need for a

security system capable of detecting reliably piggy-backing or tail-gating, whilst providing ease of use and only a minimum number of false alarms.

SUMMARY OF THE PRESENT INVENTION

According to a first aspect of the present invention a control system suitable for use with a security access device such as a security door comprises a video input device which in use captures visual image data from the secured region of the device, an image processor for processing said image data and deriving at least one parameter for discriminating the number of persons present in the secured area of the device and a comparator for comparing the said parameter derived from the original data with a predetermined threshold and producing a discriminatory output for use in controlling the interlock on the security device depending upon the result of the comparison.

The present inventors have adopted an entirely new approach to the detection of tail-gating and piggy-backing, based on the use of video data. While the use of video data has previously been proposed for the recognition of authorised persons, as an alternative, e.g., to the use of card readers, the use of video data for piggy-backing detection (APB) has not previously been thought desirable or possible. On the face of it the complexity of the visual data which would be gathered from e.g., a revolving transparent door, and the processing overheads involved in determining from such data the number of persons present provide a major disincentive to the adoption of such techniques. By contrast with recognition techniques which take place outside the door, APB detection generally has to be carried out in the short interval of time during which the user is passing through the door and so techniques involving large processing overheads tend to be avoided. However, the present inventors have realised that with appropriate processing, the raw video data can be used to yield a simple parameter for comparison with a

predetermined threshold, and that the use of video data in this manner allows effective APB detection in real time using a relatively low-powered processor. The threshold may be for a statistic derived from a number of parameters in combination.

Preferably the image processor and discriminator comprise a cascaded series of modules, each module being arranged to process image data to determine a respective parameter, and to compare the parameter with a corresponding threshold.

Preferably the modules are arranged so that when one module is able to make a decision at a predetermined confidence level, then that module produces the said discriminatory output signal, otherwise the said module passing image data on to a subsequent module for further processing.

Preferably the modules are arranged generally in order of their discriminatory power, with the most powerful module receiving the image data first.

The discriminatory power of a module is a measure of its ability to make a discriminatory decision with a minimum processing overhead, and so at maximum speed. The efficiency of the whole system is maximised by using the fastest tests firsts, and only passing on to tests with a greater processing overhead when the preceding tests fail to meet a predetermined confidence level.

Preferably the image processor is arranged to capture a background image of the door with no person present, and to capture a subsequent image of the door with a person present and to discard from the second said image video data which is unchanged from the background image.

According to a second aspect of the present invention there is provided a method of controlling a security access device, such as a security door, including capturing video image data from a secured area of the access device,

processing said image data and thereby deriving at least one parameter for discriminating the number of persons present in the secured area, and

comparing the said parameter with a predetermined threshold and producing a discriminatory output for use in controlling an interlock on the security device to lock the device when more than a predetermined number of persons are present in the secured area.

The present invention also encompasses a security access device when fitted with a control system in accordance with the first aspect of the present invention.

According to a further aspect of the present invention, there is provided a security access device including a secured region bounded by one or more wholly or partially transparent walls, characterised by a control system including a video input device arranged to view the secured region, and in that the transparent walls include filter means arranged to block or reduce the transmission of light in part of the visible/near-visible optical spectrum, and in that the video input device has a sensitivity/wavelength characteristic generally complementary to the transmission characteristic of the said filter means associated with the transparent walls, the visibility to the video input device of objects outside the transparent walls thereby being reduced or eliminated.

Revolving security doors, for example, are commonly built with glass doors. Objects outside the secured area are therefore potentially visible to any video security control system and may "confuse" any judgement made by the control system. This aspect of the present invention overcomes this problem by using walls which are transparent in one part of the visible/near-visible spectrum and a video input device which is sensitive in another part of the spectrum. The walls may, for example, be covered by a film which blocks transmission in the infra-red. The video device may either be selected to be inherently sensitive in the infra-red range, and insensitive outside that range,

or, may be provided with an input filter giving this desired sensitivity/wavelength characteristic.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a system embodying the present invention;

Figures 2a and 2b are schematic representations of captured video data at different stages of processing;

Figure 3 shows a set of pixel masks used in processing the image data to detect the head edges shown in Figure 2b;

Figure 4 is a schematic illustrating the processing of the image data to derive the image height and width;

Figure 5 is a flow diagram for the image processing in the APB control circuit;

Figure 6 is a schematic of hardware for implementing the system of Figure 1; and

Figures 7a - 7d are graphs showing experimental data obtained using the system of Figure 1.

DETAILED DESCRIPTION

An entry control system comprises a revolving door 1 which, in this example, has four compartments defined by radially extending wings 2a-2d. The movement of the wings is controlled by an interlock 3 mounted centrally above the wings. This interlock operates in a fashion conventional in revolving security doors to lock the door against movement until an appropriate control signal is received from a control unit.

A card reader 5 is connected to the door controller 4 and stands outside the door 1. In use, when someone wishes to pass through the door, that person inserts their card in the reader 5. Provided that the card is recognised, the reader 5 outputs an authorization signal to the door controller 4 which then operates the interlock 3 to free the door which is then driven by a motor through an angle sufficient for the authorised user to pass from the entrance to the exit of the door.

Such security door systems are vulnerable to misuse if there is piggy-backing, that is to say if a second

unauthorised person enters the compartment together with the authorised person, or if there is tail-gating, in which the unauthorised person passes through the door in another of the compartments at the same time as the authorised person goes through. In the present example, the control system for the door further comprises a video camera 6 which is mounted above the door looking down into an underlying compartment. The video data from the camera is processed in an APB control circuit 7 to produce a discriminatory output which goes to the controller 4 for the door to operate the interlock 3 to lock the door when piggy-backing or tail-gating is detected.

As illustrated in Figure 5, the APB control circuit has a cascaded hierarchical structure. This comprises a plurality of stages arranged generally in order of the discriminatory power of the particular parameter which is derived and used. In use, the image data is first subject to processing by the first of this hierarchy of stages. If that results in the production of a parameter having a value such that a decision on the presence or absence of a second person can be made with adequate confidence, then the processing is terminated at that stage and an appropriate output signal produced and fed to the external control unit. If, however, a decision cannot be made based on the first parameter, then the image data is passed on to the next stage in the hierarchy, for further processing and the derivation of a further parameter and so on. In this way, the system is structured to maximise the processing speed and to minimise the average processing overhead for the image data. Just a single parameter may be used for the discrimination in the majority of cases, with further parameters being derived and tested only in those marginal cases where a single parameter does not enable a decision to be made with sufficient confidence.

In the present example, the first stage of the video control circuit is a background-subtraction stage. This firstly captures a background image of the compartment of

the door prior to the entry of the user into the field of view of the camera. This background image is stored in order to provide a means of identifying changes occurring when entry into the door compartment is effected. Subsequently, as the user passes into the field of view, the system captures a second image, the "present" image. This is compared with the stored background image and those pixels which are unchanged are discarded, thereby eliminating them from the subsequent processing stages. In this example, the comparison is carried out by taking the grey scale values in the range 1-255 of each pixel, subtracting one from the other and writing a value of 0 for each pixel where the result of the subtraction is 0 or is less than a predetermined small value, say 10. Where the difference is greater than this predetermined threshold, then the value written is simply the grey scale value of the present image for that pixel. In the course of this process a count is maintained of the number of non-zero pixels produced and this is used as the first test parameter for APB detection. It is found that when two or more persons are present the difference count is characteristically higher than when a single person is present in the door. In some cases however the result of the difference count will be too close to the relevant predetermined threshold for a reliable determination to be made. In this case the video data is passed on for subsequent stages of processing and discrimination.

Referring again to Figure 5, in this example the next stage carries out edge detection using the known Sobel edge detection algorithm. This algorithm is applied only to the non-zero pixels of the image output from the first stage and sets those pixels to one of two binary values producing a binary image with, for example, edges shown in black on a white ground as in Figure 2a. A determination is then made of the area inside the edges. This is done by stepping a window of dimensions, e.g., 10 x 10 through the image and recording a count for each pixel which is inside

the edges and for which none of the pixels of the window cross an edge. This count then is a measure of the area inside the edges and is related to the area in the view of the camera of the shoulders and head of the person passing through the door. This measure of the area provides a second discriminatory parameter which as in the first stage is compared with a respective discriminatory threshold. As before, if this comparison produces a result with a sufficient degree of certainty, then the processing can be terminated at that stage and the discriminatory output passed on to the control system 7. If this is not the case, then the system passes on to the next stage of processing. As illustrated in Figure 4, this calculates the height and width of the image defined by the edges, calculates the ratio of the two, and compares this with a respective predetermined threshold.

In this example the final processing module carries out what has been termed "head edge detection". This uses the aliasing characteristic of the "corners" of the detected edges. The image data is compared with a set of pixel masks of the form shown in Figure 3. A count is incremented, and a mark displayed (Figure 2b), for each pixel where a match is found with one of the masks. The count of the number of matches is then compared with a predetermined threshold for the count to provide a further discriminatory test.

When any of the tests produces an output indicating that two or more people are in the door, then the control system 7 may activate the interlock to lock the door against movement, thereby trapping the users, or may drive the door in the reverse direction to expel the users.

The inventors have found that the discrimination of the system can be improved by calculating an additional parameter to compensate for the position of an object in the door. The parameters discussed above are based on what is assumed to be a plan view from above of the person in the door. However, if the person is not directly under the

camera, then the camera captures part of the side of the body too. This extra information would tend to distort the measurement and could lead to the mis-classification of a tall off-centre person as two people. To avoid this, a measurement is taken first of the position of the centre of the object. This is then compared with the actual centre of the camera and an adjustment parameter is set. The adjustment parameter takes the form of a ratio by which subsequently determined parameters are multiplied.

Figure 7a and Column 1 of Table 1 show the measurements of the position of the object used for determining the adjustment ratio for a sample of 72 test images. The higher the number the further the object is from the centre of the camera. The images are a selection from the following categories: a single person; two people; people carrying large and small objects; a person carrying a large object over their head.

Figure 7b and Column 2 of the table show the difference count parameter, after correction by the adjustment ratio or "normalising factor" of Figure 7a. Note that images 68-70 are large objects carried over the top of the head and so result in a large difference count. Figure 7c and Column 3 show the (corrected) head area parameter. This measurement is a guide to the total area available to be a head space. Images 68-70 would cause concern as the values are extremely high. These images would fall above the alarm threshold, as 2 people could fit under such a large object. Figure 7d shows the height/width ratio parameter measuring the compactness of the image. The lower the value, the more likely that the imaged object is two people.

Table 1 below lists the data illustrated in these graphs. Columns 1 to 3 are identified above. The data contained in the other columns are as follows:

Column 4 Width

This is the width of the Head Area region.

Column 5 Height
This is the height of the Head Area region.

Column 6 Normalised Ratio
This is the ratio of the width and height. The Normalising Factor is used to adjust the ratio depending on the position of the object in respect to the centre of the camera.

Column 7 Head Edges
This measurement is the number of pixel patterns that match with a head indicator mask. The greater the number the larger the number of head mask matches.

Column 8 Head Edge Groups
This is the number of concentrated groups of Head Edges.

Column 9 Straight Edge Indicator
This measurement is the number of straight lines found in the image.

Column 10 Volume
This is the total volume that would encapsulate the object.

Column 11 Volume Width
This is the width of volume.

Column 12 Volume Height
This is the height of volume.

Column 13 Volume Ratio
This is the ratio of the volume height and volume width.

Not all of the parameters listed in the table need be used in any given implementation of the invention. The comparators may be arranged to make an "intelligent"

decision based on a number of parameters, rather than simply applying a single threshold. The system may also include a knowledge base so that it can judge each situation on the measurements of the current image and knowledge gained about the environment the system is working with.

Table 2 lists the subjects used for the different images listed in Table 1.

For the data of this example, the following thresholds may be used by the comparators for discrimination:

<u>Normalised Difference</u>

Below 750 - Allow entry

Above 15000 - Refuse entry. Suspect door entry

<u>Area</u>

Below 750 - Allow entry

Above 4500 - Refuse entry. Suspect door entry

<u>Normalised Ratio</u>

Above 95 - Allow entry

Not all measurements have a pass & fail level. For example if the ratio is above 95 then we would allow entry. If it is below this level, we use other measurements to make a decision.

The system described above may be modified through the use of moving images, that is to say the capturing of a series of video frames as the person moves through the door. This then produces distinctive "constants" which may be used as the basis for discriminatory parameters in an analogous fashion to the process discussed above.

In systems embodying the invention, some of the processing modules may use appropriately trained neural networks. These may be trained on the parameters produced by other modules.

Appropriate thresholds for the different parameters used by different modules may be obtained by inspection from real sample data such as that illustrated in Figures

7a-7d, and or from computer modelling of the environment in which the system is to be used.

Figure 6 illustrates one example of hardware embodying the system. A camera 61 views the target scene and passes the acquired image to a frame grabber 62. In this preferred implementation, the camera views the target scene through an IR bandpass filter. An IR bandstop filter which may be in the form of a film applied to the walls of the revolving door and which is transparent in the rest of the visual spectrum is then used to screen the extraneous visual field from the field of view of the camera. A light source may be provided inside the door to provide illumination at levels appropriate to the sensitivity of the video camera 61.

The image from the frame grabber is passed to a single board PC 63 which runs image processing software implementing the hierarchical modular structure discussed above. Once a decision has been made by the control program, an output is passed to the door controller 64.

In the presently described example, the camera 61 is a 1/3 inch black and white EI camera (RS845-184) having a 1/3 inch CS mount DD lens, 2.6mm (RS846-266). The frame grabber is that available commercially as VIDEOBLASTER SE. In this example the single board PC uses an Intel 486DX2/66 microprocessor and an ISA bus. The interface to the door controller 64 is a standard PC ISA bus serial card. The user interface which enables the user to set adjustable parameters for the control program may comprise a keyboard or keypad and a VDU or LCD display.

It will be understood that the above components are given by way of example only, and that a number of alternative implementations are possible.

The following appendices list, in appendix A, the source code for the image processing/discrimination program run on the single board PC 63, and, in appendix B, the different elements of the program are listed in pseudocode.

TABLE 1 (i)

IMAGE	C1	C2	C3	C4	C5	C6	C7	C8	C9 - COLUMN
1	12.08305	5724	1531	93	132	103	120	35	103
2	8.602325	4742	807	86	78	120	150	53	103
3	12.96148	5163	1395	94	99	142	115	28	130
	12.56981	8002	2537	110	131	124	165	30	145
	2.828427	6888	1037	80	102	86	140	35	97
	9.273618	3800	923	65	76	116	77	22	85
	6.480741	4237	387	87	91	119	114	26	80
	10.48809	2725	771	56	74	106	77	24	45
	12.16553	4640	1199	96	98	144	154	45	103
10	4.690416	4605	217	86	93	109	134	28	69
	4	9708	1269	92	123	85	252	90	127
	7.348469	9961	1978	92	140	84	192	33	135
	5.830952	12648	1999	184	100	66	270	88	175
	4.242641	10147	1616	89	120	86	255	78	143
	10.95445	9424	3431	105	156	96	192	57	123
	11.6619	8120	1771	134	147	133	229	63	140
	12.16553	6770	2903	109	104	141	122	36	80
	12.24745	6353	2149	108	102	140	167	46	113
	4.242641	7262	403	190	92	56	295	87	82
20	6.164414	12320	4322	118	170	86	238	72	172
	5.477226	7536	2034	164	96	70	151	28	103
	8.3666	7090	2287	118	129	121	157	43	90
	5.830952	7082	2571	121	72	72	214	52	72
	7.071068	6891	1988	93	91	124	170	41	91
	4	10270	2772	145	143	113	134	35	163
	5.291503	10284	2290	155	138	107	206	65	127
	5.656854	4874	879	82	83	120	193	65	78
	11.48913	5908	2037	98	124	115	138	38	129
	10.95445	6081	1663	130	108	119	170	54	126
30	5.291503	6756	2111	92	123	89	133	38	117
	9.69536	5352	1621	81	103	108	151	31	87
	6.480741	6925	2899	123	103	104	157	37	92
	4	6862	2324	94	111	97	197	53	69
	4	9821	3078	112	132	97	240	98	135
	8.124038	11976	3076	126	157	105	228	55	181
	4.242641	13678	1678	118	171	80	511	169	139
	2.828427	5252	978	77	82	103	106	22	73
	4.472136	9423	2567	94	138	80	232	61	146
	5.656854	8546	1243	106	143	90	195	59	140
40	3.464102	6145	1203	128	80	70	173	44	81
	2.828427	5796	1261	132	79	65	151	33	86
	3.464102	6516	1725	155	84	61	134	36	86
	6.324555	6073	537	107	87	101	173	48	105
	5.830952	8056	2271	129	105	99	189	46	128
	4.690416	9984	2531	117	138	99	303	112	117
	5.656854	1383	715	79	46	71	39	10	43
	8.717798	819	1006	45	40	118	41	10	21
	12.56981	-925	82	28	32	130	24	6	28
	11.13553	8370	3984	158	101	91	175	38	122
50	10.09951	5835	2215	127	116	127	185	50	102
	9.165151	5289	2195	116	98	114	136	34	90
	11.74734	6256	3532	138	99	104	139	37	86
	8.124038	7045	2126	88	106	109	202	53	118
	12.80625	4969	2527	81	90	136	149	53	58
	11.91638	5228	2251	96	95	144	172	40	106

TABLE 1 (ii)

IMAGE	C1	C2	C3	C4	C5	C6	C7	C8	C9 - COLUMN
	9.165151	5904	2249	82	72	118	201	86	63
	12.32883	5811	3225	104	99	141	157	41	73
	11.83216	8789	4192	102	156	95	198	60	148
	11.40175	5393	3357	99	100	142	136	48	66
60	10.77033	4947	2292	104	76	104	138	31	68
	5.477226	6150	2355	130	66	60	265	154	69
	9.591663	4957	2422	113	85	103	112	32	77
	9.380832	5654	2489	127	114	122	140	37	92
	10.19804	4186	2294	83	70	118	122	43	61
	9.797959	5108	2359	96	89	128	120	42	83
	5.830952	6286	2231	99	105	115	150	28	99
	15.42725	6134	3250	115	116	160	144	52	98
	12.49	18356	15000	163	187	130	144	35	100
	8.246211	28437	16666	219	254	114	266	46	177
70	10.09951	24673	15205	214	254	117	234	50	215
	6.928203	6718	967	123	116	120	259	83	111
	2.828427	7376	1401	115	106	102	197	66	114

TABLE 1 (iii)15

TMAC	C10	C11	C12	C13 - COLUMN
1	3255	85	34	59
2	3010	76	34	59
3	4700	85	49	86
•	7480	93	68	109
•	4080	76	50	72
•	2470	62	37	80
	3567	73	40	67
	1443	47	37	110
	4800	74	49	98
10	4214	73	48	77
	4895	89	54	69
	6048	88	71	103
	8050	156	45	34
	5340	87	60	79
	7313	98	70	102
	9916	112	73	95
	3763	90	52	84
	4550	99	49	73
	6222	145	33	25
20	7600	98	79	99
	7872	115	48	49
	5310	114	45	52
	4235	108	34	38
	3534	80	37	59
	6448	117	52	51
	8970	144	64	53
	3116	69	37	64
	5782	88	58	94
	3936	104	48	66
30	4402	80	61	92
	4212	81	51	86
	6273	123	51	51
	4968	90	53	67
	5400	100	59	67
	6237	97	77	104
	8455	106	88	97
	2541	71	33	51
	6510	86	69	94
	7632	104	71	83
40	4148	92	33	39
	5280	94	39	45
	4662	110	36	36
	4494	96	41	52
	6837	129	52	49
	6786	89	57	76
	1738	57	22	46
	945	45	20	59
	384	23	16	103
	7050	137	47	49
50	6985	102	55	74
	3045	93	34	49
	5292	118	41	49
	3960	81	44	71
	3432	79	44	83
	3360	95	34	51

TABLE 1 (iv)

	C10	C11	C12	C13
	2414	76	33	58
	3478	99	37	55
	6794	92	78	123
	2844	85	35	59
60	2992	96	34	50
	3564	119	33	32
	4104	97	38	53
	4080	108	33	41
	2822	83	33	54
	3268	86	42	66
	2904	76	43	69
	3948	96	42	69
	15040	149	93	92
	24852	213	113	70
70	24282	201	114	78
	6273	117	50	53
	5175	112	45	44

TABLE 2

IMAGE NO.	SUBJECT
1 to 5	1 person -positional test
6 to 10	1 person -positional test
11 to 14	2 people -positional test
15 to 19	1 person-positional test
20	2 people
21 to 24	1 person + brolly
25 to 26	1 person + box
27 to 29	1 person -positional test
30 to 33	1 person + brolly
34 to 36	1 person + box
37	1 person + brolly
38	2 people
39	1 person + box
40 to 42	1 person + brolly
43	1 person + briefcase
44	1 person + brolly
45	2 people
46 to 48	postbag in door
49	small single object
50	1 person + rucksack
51-57	1 person carrying object
58-60	1 person +hooded coat
61-68	1 person -different arm positions
69-71	1 person - box overhead
72	1 person -hat

APPENDIX A


```
//**********************************************************
/*      Mayor - Guard - Door Secuirty Program              *
/*                                                         *
/*                                                         *
/*      Martin Golding                                     *
/*      mg@ukc.ac.uk                                       *
/*      Written in C using Turbo Borland C                 *
//**********************************************************


// these include other software needed to run the program
#include <ctype.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>

// the size of the images that we use. Can be modified latter to
speed
// up opreation
int const IMAGE_SIZE = 256;

// these are used to hold the image imformation
unsigned char ** Img1;
unsigned char ** Img2;
unsigned char ** Back;

//**************************************************
// creates space for a new image in memory
//
void newImage(unsigned char * ** i,int size) {

     (*i) = new unsigned char * [size];

     for (int x=0;x<size;x++) {
          (*i)[x] = new unsigned char[size];
          if (!((*i)[x])) {
               clrscr();
               cout << "\nImage Error\n\n";
               getch();
               exit(1);
          }
     }
}
//**************************************************
// deletes space taken up by an image
//
void deleteImage(unsigned char*** i,int size) {
     for (int x=0;x<size;x++) {
          if (((*i)[x])) delete ((*i)[x]);
     }
     delete (*i);
}
//**************************************************
```

```
// displays the image on the screen
//
void display(unsigned char** Img) {
int Size = IMAGE_SIZE;

          for (int y = 0; y < Size; y++)
          for (int x = 0; x < Size; x++)
               putpixel(x,y,Img[y][x]/16);
}
//****************************************************
// creates all the space for the images required
//
void start() {
     newImage(&Img1,256);
     newImage(&Img2,256);
     newImage(&Back,256);
}
//****************************************************
/ deletes all the space for the images
void end() {
     deleteImage(&Img1,256);
     deleteImage(&Img2,256);
     deleteImage(&Back,256);
}
//****************************************************
// turns the computer into graphics mode so we can see the image
//
void startGraphics() {

     int errorcode, graphdriver, graphmode;

     if (registerfarbgidriver(EGAVGA_driver_far) < 0) exit(1);

     graphdriver     = VGA;
     graphmode       = VGAHI;

     initgraph(&graphdriver, &graphmode, ".." );
     errorcode = graphresult();
     if (errorcode != grOk) {
          cout << "Cant do graphics. \n" ;
          exit(1);
     }
     struct palettetype pal;
     getpalette(&pal);

     // create gray scale
     for (int i = 0; i < pal.size; i++)
     setrgbpalette(pal.colors[i], i*4, i*4, i*4);
}
//****************************************************
// resets the computer from graphics mode
//
void endGraphics() {
     closegraph();
}
//****************************************************
// use roberts operator to detect a presence of a line
//
```

```
void edgeDetect(unsigned char** imgIn,unsigned char** imgOut,int
level) {


       int ySize = IMAGE_SIZE;
       int xSize = IMAGE_SIZE;

       double a,b,c,d,rob;

       for(int y=0;y<ySize-1;y++)
       for(int x=0;x<xSize-1;x++) {

            // only look a pixel that is not background
            if (imgIn[y][x]) {

                a = double(imgIn[y][x]);
                b = double(imgIn[y][x+1]);
                c = double(imgIn[y+1][x]);
                d = double(imgIn[y+1][x+1]);

                rob = sqrt((((a-d)*(a-d))+((b-c)*(b-c))));

                if (rob>=(double)(level))  // if a line
                                imgOut[y][x] = 255;
                           else
                                // not
                                imgOut[y][x] = 1;

            } else imgOut[y][x] = 0;
      }

}
//****************************************************
//  realign all black pixels to be 1 instead of 0 so we speed up
operation.
//  All the algorithms know a pixel of 0 is not worth looking at.
//
void levelBlack(unsigned char** in) {

      for(int y=0;y<IMAGE_SIZE;y++)
      for(int x=0;x<IMAGE_SIZE;x++)
           if (in[y][x]==0) in[y][x] = 1;
}
//****************************************************
// set all the image to be 0
//
void blank(unsigned char** in) {

      for(int y=0;y<IMAGE_SIZE;y++)
      for(int x=0;x<IMAGE_SIZE;x++)
           in[y][x]=0;
}
//****************************************************
// deduct the background image from the present image and save
the result
//
long deduct(unsigned char** back, unsigned char** img, unsigned
char** out,int LEVEL) {
```

```
 int value ;
 long count=0;
      int ImageSize = IMAGE_SIZE;

      for(int y=0;y<ImageSize;y++)
      for(int x=0;x<ImageSize;x++) {
           value = (int)(img[y][x]);

                // if the absolute value is greater the level
given
                // then set the out value
                if (abs((int)(back[y][x])-value)>LEVEL) {
                     out[y][x] = img[y][x] ;
                     count++;
                } else
                     // otherwise turn to not worth looking at
                     out[y][x] = 0;
      }

// return the number of differences
return(count);
}
//**************************************************
// read in the image file named with 'inFile'
//
int ReadInFile(unsigned char** Img, char *inFile) {

     FILE *file;
     int ch;

     if ((file=fopen(inFile,"rb"))==NULL) {
               cout << "Cant open file "<< inFile <<"\n";
               return(0) ;
     }

     char Header[512] ;
     fread(Header,1,512,file);

     for (int y=0;y<IMAGE_SIZE;y++)
     for (int x=0;x<IMAGE_SIZE;x++)
          if ((ch=fgetc(file))==EOF) {
               cout << "Reached end before should have\n";
               fclose(file);
               return(0);
          } else {
               Img[y][x] = ch ;
          }

     fclose(file);

     return(1) ;

}
//**************************************************
// get the details of the image file
//
int GetFileInfo(char *inFile, int* grays, int* size) {
```

```
        FILE *file;

        if ((file=fopen(inFile,"rb"))==NULL) {
                cout << "Cant open " << inFile << "\n";
                return(0);
        }

        char Header[43] ;

        fread(Header,1,43,file) ;

        int GrayLevels ;
        int ImageRes ;

        div_t temp ;

        GrayLevels = (int)Header[41] ;
        GrayLevels = (1 << GrayLevels) ;
        temp = div(512,(int)Header[42]) ;
        ImageRes = temp.quot ;
        *grays = GrayLevels;
        *size = ImageRes ;

        fclose(file) ;

        if (GrayLevels==256) {
        if (ImageRes==IMAGE_SIZE) {
                return(1);
        } else {
                return(0);
        };
        } else {
                return(0);
        };

}
//*****************************************************
// check the image file is ok and import the file
//
int ImportFile(unsigned char** Img, char* inFile) {

        int GrayLevels;
        int ImageSize;

        if (!GetFileInfo(inFile,&GrayLevels,&ImageSize)) {
                return(0);
        }

        if (!ReadInFile(Img,inFile)) return(0);

        return(1);

}
//*****************************************************
// export the image into a file
//
int ExportFile(unsigned char** Img, char* name) {
```

```
        FILE *file;
        static char ext[1];
        ext[0]='\0';

         if ((file=fopen(name,"rb"))!=NULL) {
         {
                  fclose(file);
                  remove(name);
         }
         }

        if ((file=fopen(name,"wb"))==NULL) {
        {
             printf("Cannot save... press any key to cont");
             getch();
        }
        }
        char Header[512] ;

        int ImageRes ;

        div_t temp ;

        ImageRes = IMAGE_SIZE;

        Header[41] = 8;

        temp = div(512,ImageRes) ;
        Header[42] = temp.quot;

        fwrite(Header,1,512,file) ;

        int imgSize = IMAGE_SIZE;

        int Val;

        for(int y=0;y<imgSize;y++)
             for(int x=0;x<imgSize;x++) {
                  Val=Img[y][x];
                  fputc(Val,file);
             }

        fclose(file) ;
return(1);
}
//****************************************************
// look around the image and mark areas that could be big enough
// for a head. Using a window of 10x10
//
int FindPossibleHeadSpace(unsigned char** in,unsigned char**
out,int look=1) {
int const WIN = 10;
int count;
int result = 0;

      for(int y=0;y<IMAGE_SIZE-WIN;y++)
      for(int x=0;x<IMAGE_SIZE-WIN;x++)  {
```

```
                // if 1 should look at this pixel, 0 means dont bother
                if (in[y+WIN/2][x+WIN/2]) {
                     count =0;

                     // count all the ones to look for
                     for(int yy=y;yy<y+WIN;yy++)
                     for(int xx=x;xx<x+WIN;xx++) {
                          if (in[yy][xx]==look) {
                               count ++;
                          } else
                               break;
                     }

                     // if they are all to be looked at
                     if (count==WIN*WIN) {

                          // fill them all in
                          for(int yy=y;yy<y+WIN;yy++)
                          for(int xx=x;xx<x+WIN;xx++) {
                               out[yy][xx] = 255;
                          }
                          result++;

                     }
                }

          }

     // return all the ones found
     return(result);
     }
     //***************************************************
     // detects the head edges using head masks
     //
     int DetectPossibleHeadEdges(unsigned char** img, unsigned char**
     imgOut) {
     int iSize = IMAGE_SIZE;
     int x,y,xx,yy,cnt=0;
     int x1,x2,x3,x4,x5,x6,x7,x8,x9;

     // go around in a 3x3 window
     for(x=1;x<iSize-3;x++)
     for(y=1;y<iSize-3;y++)

     // dont bother looking at a 0
     if (img[y][x]) {

          // get all 9 elements
          x1=img[y-1][x-1];
          x2=img[y-1][x];
          x3=img[y-1][x+1];
          x4=img[y][x-1];
          x5=img[y][x];
          x6=img[y][x+1];
          x7=img[y+1][x-1];
          x8=img[y+1][x];
          x9=img[y+1][x+1];
```

```
        // look at all the 6 masks
             // if found then increase count
if
((x1)&&(!x2)&&(!x3)&&(x4)&&(x5)&&(!x6)&&(x7)&&(x8)&&(!x9))
{ imgOut[y][x] = 255; cnt++; }
else if
((!x1)&&(!x2)&&(x3)&&(!x4)&&(x5)&&(x6)&&(!x7)&&(x8)&&(x9))
{ imgOut[y][x] = 255; cnt++; }
else if
((!x1)&&(!x2)&&(!x3)&&(!x4)&&(x5)&&(x6)&&(x7)&&(x8)&&(x9))
{ imgOut[y][x] = 255; cnt++; }
else if
((!x1)&&(!x2)&&(!x3)&&(x4)&&(x5)&&(!x6)&&(x7)&&(x8)&&(x9))
{ imgOut[y][x] = 255; cnt++; }
else if
((x1)&&(x2)&&(x3)&&(x4)&&(x5)&&(!x6)&&(!x7)&&(!x8)&&(!x9))
{ imgOut[y][x] = 255; cnt++; }
else if
((x1)&&(x2)&&(x3)&&(!x4)&&(x5)&&(x6)&&(!x7)&&(!x8)&&(!x9))
{ imgOut[y][x] = 255; cnt++; }


}

// return the numbers of head edges there are
return(cnt);

}
//***************************************************
// find the height of the object
//
int height(unsigned char** in) {
int x,y;
int start=-1,end=IMAGE_SIZE+1;

     for(y=0;y<IMAGE_SIZE;y++)  {
          for(x=0;x<IMAGE_SIZE;x++)  {
               if (in[y][x]==255) {
                    start=y;
                    break;
               }
          }
          if (start!=-1) break;
     }

     for(y=IMAGE_SIZE-1;y>=0;y--)  {
          for(x=0;x<IMAGE_SIZE;x++)  {
               if (in[y][x]==255) {
                    end=y;
                    break;
               }
          }
          if (end!=IMAGE_SIZE+1) break;
     }

if (start==-1) return(0);

// return the height
```

```
return(end-start);

}
//**************************************************
// find the width of the object
//
int width(unsigned char** in) {
int x,y;
int start=-1,end=IMAGE_SIZE+1;


     for(x=0;x<IMAGE_SIZE;x++)  {
          for(y=0;y<IMAGE_SIZE;y++)  {
               if (in[y][x]==255) {
                    start=x;
                    break;
               }
          }
          if (start!=-1) break;
     }

     for(x=IMAGE_SIZE-1;x>=0;x--)  {
          for(y=0;y<IMAGE_SIZE;y++)  {
               if (in[y][x]==255) {
                    end=x;
                    break;
               }
          }
          if (end!=IMAGE_SIZE+1) break;
     }

if (start==-1) return(0);
//return the width
return(abs(end-start));

}
//**************************************************
// the main program

int main(int, char* []) {


start(); // start up the space for the images
startGraphics(); // turn on the graphics

ImportFile(Back,"back.img");  // load up the background image

display(Back); // display and wait
getch();

ImportFile(Img1,"frame.img"); // load up the present image

display(Img1); // display and wait
getch();

levelBlack(Img1); // realign the black pixels so we can use them
levelBlack(Back); // to know what to look for
```

```
long diff = deduct(Back,Img1,Img2,17); // take the background
from
                                   // forground and get the difference

display(Img2); // display the result and wait
getch();

edgeDetect(Img2,Img1,30);  // detect the edges in the image

display(Img1); // display the result and wait
getch();

blank(Img2); // set the image to all 0s

int area = FindPossibleHeadSpace(Img1,Img2); // get the total
area

display(Img2); // display result and wait
getch();

int h = height(Img2); // calculate the height
int w = width(Img2);  // calculate the width

blank(Img1); // set the image to all 0s

int edegs = DetectPossibleHeadEdges(Img2,Img1); // look for
possible head edges

display(Img1); // display result and wait
getch();

endGraphics();  // shut down the graphics
end(); // release all the memory for the images

cout << "diff\t" << diff << endl; // output the results
cout << "area\t" << area << endl;
cout << "width\t" << w << endl;
cout << "height\t" << h << endl;

double ratio;

if (w>h)  // calculate the ratio of height/width
     ratio =  double(h)/double(w);
else
     ratio =  double(w)/double(h);

cout << "ratio\t" << ratio << endl;
cout << "edges\t" << edges << endl;


double const RATIO = 0.6; // define the thresholds
int const EDGE = 45;
long const DIFF = 10000;

cout << endl << endl;

     // tell them what you are looking for
cout << "We will look for a difference of " << DIFF << endl;
```

```
cout << "We will look for a edges of " << EDGE << endl;
cout << "We will look for a ratio of " << RATIO << endl;

cout << endl << endl;


// see how we have done

     if (diff<DIFF) {
               cout << "one person  -  on differences" << endl;
     } else {
               cout << "two people  -  on differences" << endl;
     }
     if (edges>EDGE) {
          cout << "two people  -  on edges" << endl;
     } else {
          cout << "one person  -  on edges" << endl;
     }

     if (ratio<RATIO) {
          cout << "two people  -  on ratio" << endl;
     } else {
          cout << "one person  -  on ratio" << endl;
     }

return(1); // finished

}
```

APPENDIX B

```
//*****************************************************
//*   Mayor - Sydo Guard - Door Security Program     *
//*                                                  *
//*                                                  *
//*                                                  *
//*   Martin Golding                                 *
//*   mg@ukc.ac.uk                                   *
//*   Written in sydo-code                           *
//*****************************************************


//****************************************** define the size of the
image to be 256

define the number of gray scales as 256

define a pixel of value 0
        A pixel value of 0 will be given to a pixel that is not to
        be looked at and not used in the evaluation.

define a pixel in the range 1-255
        A pixel in this range will be looked at and will be used in
  the evaluation.

define WHITE equals 255.

define BLACK equals 1.

define NOTHING equals 0.

//******************************************
Main part of the program  start
        CreateImage         Create space for images in the computer.

        ImportBackGround    Get the background image.
```

```
        ImportCurrentImage Get the current image that we need to
test for
                           people.

        Level0OutOfImage   Convert the background image so it does
not have
        BackGround         any 0 value pixels.

        Level0OutOfImage   Convert the current image so it does not
have
        Current            any 0 value pixels.

        RemoveBackgroud    Remove the background from the current
image.
                           Mark all pixels that are not part of the
person/s
                           that have moved into view with a 0
value. The
                           number of differences are assigned to
the
                           DIFFERENCE_VALUE variable. Use a
threshold
                           level for deduction.

        Decide if value DIFFERENCE_VALUE is strong enough to make a
        decision.
                YES - decide
                NO  - carry on

        EdgeDetect         Detect all the edges of the image. Use a
threshold
                           level for possibility for an edge.

        FindAreaForHead    Find areas within the image that could
possibly
                           be big enough for a head. The number of
areas
                           found is assigned to the AREA_VALUE
variable.

        Decide if value AREA_VALUE is strong enough to make a
        decision.
                YES - decide
                NO  - carry on

        CalculateHeight    Calculate the height of the areas found.
The
                           height of the areas is assigned to the
HEIGHT_VALUE
              variable.

        CalculateWidth     Calculate the width of the areas found.
The width
                           of the areas is assigned to the
WIDTH_VALUE
                           variable.

        CalculateRatio     Use the HEIGHT_VALUE and the WIDTH_VALUE
to get
                           a ratio of the area found. The ratio
value is
              assigned to the RATIO_VALUE variable.

        Decide if value RATIO_VALUE is strong enough to make a
        decision.
```

YES - decide
NO - carry on

DetectHeadEdges Detect edges that could be part of a person. The number of person edges found is assigned to the HEAD_EDGES_VALUE variable.

Decide if value HEAD_EDGES_VALUE is strong enough to make a decision.
YES - decide
NO - carry on

....
....
....

This section can be either continued with further processing, re-tried with different thresholds or a best guess can be done.

....
....
....

DeleteImage Deletes space for images in the computer.

end Main part of the program.
//**
The next section will define all the blocks above.
//**
CreateImage Input Values : A pointer to the image. The size of the image needed.
Output Result : A pointer to the space allocated for the image.

allocate enough pointers for the columns.

at each column allocate enough space for the rows.

//**
DeleteImage Input Values : A pointer to the image. The size of the image.
Output Result : A pointer to the no space.

delete space in each columns so we delete all the rows.

delete columns pointers.

//**
ImportFile Input Values : A pointer to the image. The name of the file.
Output Result : The image is either placed in the space for the image or an error occurs.

Get the information from the file. Make sure it is correct in size and the number of gray scales.

if not correct then produce an error.

Else then read in the file in the space provided.

```
//*****************************************
ImportBackGround    Input Values   : The name of the file.
Output Result : The image is either placed in the space for the
image
                or an error occurs.

        run ImportFile( backGround , background filename).

//*****************************************
ImportCurrentImage  Input Values   : The name of the file.
Output Result : The image is either placed in the space for the
image
                or an error occurs.

        run ImportFile( currentImage , currentImage filename).

//*****************************************
Level0OutOfImage    Input Values   : The image we will work on.
Output Result : The image will have no 0 value pixels in it.

        start at the top left hand corner pixel.

LOOP_1: get the image value.

        is value equal to 0
                yes : replace pixel value with a BLACK.

        move to the next pixel on the right.

        are we on the right hand edge of the image.
                yes : move down one pixel, go back to the
                      first lefthand pixel.

        are we on the bottom edge of the image.
                no : goto LOOP_1.

        we have completed the process.

//*****************************************
RemoveBackgroud      Input Values   : The background, the
currentImage,
                the result image, a threshold level.
Output Result : The result image will contain a image that is the
current
                image with the background removed.
                The number of differences there are will be given.

        start at the top left hand corner pixel.
        start counting the number of differences at 0.

LOOP_1:
        get the background image pixel value.
        get the current image pixel value.

        if the difference between these 2 pixels greater than the
threshold.
                yes:
                        copy the current image pixel into the
result.
                        image pixel
                        add one to the number of differences.
                no:
                        let the result image pixel equal NOTHING.
```

```
        move to the next pixel on the right.

        are we on the right hand edge of the image.
                yes : move down one pixel, go back to the
                      first lefthand pixel.

        are we on the bottom edge of the image.
                no : goto LOOP_1.

        we have completed the process.
        return the value of the number of differences.

//*****************************************
EdgeDetect          Input Values  : The image to work on, the
resulting image, a threshold level.
Output Result : The result image will contain the edges of the
current image.

        start at the top left hand corner pixel.

LOOP_1:
        get the image pixel value.

        is the pixel value equal to NOTHING.
                no:
                        copy the current image pixel into the
result.
                        let a = current pixel value.
                        let b = next left pixel value.
                        let c = next down pixel value.
                        let d = next left and down pixel value.
                        line value = square root( square(a-d) +
square(b-c) ).
                        if line value is greater then the
threshold.
                                yes:
                                        set result pixel equal to
WHITE.
                                no:
                                        set result pixel equel to
BLACK.

                yes:
                        set result image pixel to NOTHING.

        move to the next pixel on the right.

        are we on the right hand edge of the image minus 1 pixel.
                yes : move down one pixel, go back to the
                      first lefthand pixel.

        are we on the bottom edge of the image minus 1 pixel.
                no : goto LOOP_1.

        we have completed the process.

//*****************************************
FindAreaForHead     Input Values  : The image to work on, the
result image.
Output Result : The result image will contain areas marked
                that are possibly large enough for body features.
                Will return the number of areas found.

        define a the sub-window size to be 10 called WIN_SIZE.
```

```
        start at the top left hand corner pixel.
        start counting the number of areas found at 0.

LOOP_1:
        get the image pixel value.

        is the pixel equal to NOTHING.
                no:
                        look around the current pixel in a WIN_SIZE
                        window. Count all the pixels that are
BLACK.

                        if the count was equal to WIN_SIZE*WIN_SIZE
                        then all pixels in the window were BLACK.

                        count equal to WIN_SIZE*WIN_SIZE.
                        yes:
                                set all the window of WIN_SIZE
around
                                current pixel in the result image
to
                                be WHITE.
                                set the area count to be plus 1.

        move to the next pixel on the right.

        are we on the right hand edge of the image.
                yes : move down one pixel, go back to the
                      first lefthand pixel.

        are we on the bottom edge of the image.
                no : goto LOOP_1.

        we have completed the process.
        return the value of the number of areas found.


//****************************************
CalculateHeight     Input Values   : The image to work on.
Output Result : The height found in the image.

        go from the top of the image to the bottom, stop at the
first
        WHITE pixel found.

        go from the bottom of the image to the top, stop at the
first
        WHITE pixel found.

        we have completed the process.
        return the value of the 2 numbers taken from each other.

//****************************************
CalculateWidth      Input Values   : The image to work on
Output Result : The width found in the image

        go from the left of the image to the right, stop at the
first
        WHITE pixel found.

        go from the right of the image to the left, stop at the
first
        WHITE pixel found.

        we have completed the process.
```

```
        return the value of the 2 numbers taken away from each
other.

//****************************************
CalculateRatio        Input Values   : The width value, the height
value.
Output Result : the ratio value.

                  if width is greater than the height.
                       yes:    divide height by width.

                  if height is greater than the width.
                       yes:    divide width by height.

        we have completed the process.
        return the ratio value.
//****************************************
TestMaskWithWindow
Input Values   : The mask value, the window pixel values to look at.
Output Result  : True is the mask works for the pixel value False
                 if not.

        blanks means a NOTHING value.
        'o' means a WHITE value.

3x3     Each element in the mask represents a comparision with the
        window.

mask1
-----

        +---+---+---+
        | o |   |   |
        +---+---+---+
        | o | o |   |
        +---+---+---+
        | o | o |   |
        +---+---+---+

mask2
-----

        +---+---+---+
        |   |   | o |
        +---+---+---+
        |   | o | o |
        +---+---+---+
        |   | o | o |
        +---+---+---+

mask3
-----

        +---+---+---+
        |   |   |   |
        +---+---+---+
        |   | o | o |
        +---+---+---+
        | o | o | o |
        +---+---+---+

mask4
-----

        +---+---+---+
        |   |   |   |
```

```
        +---+---+---+
        | o | o |   |
        +---+---+---+
        | o | o | o |
        +---+---+---+

mask5
-----
        +---+---+---+
        | o | o | o |
        +---+---+---+
        | o | o |   |
        +---+---+---+
        |   |   |   |
        +---+---+---+

mask6
-----
        +---+---+---+
        | o | o | o |
        +---+---+---+
        |   | o | o |
        +---+---+---+
        |   |   |   |
        +---+---+---+

//*******************************************
DetectHeadEdges     Input Values   : The image to work on, the
result image.
Output Result : The result image will contain points marked that
                are possible points on a body.
                The number of points marked as head edges.

        start at the top left hand corner pixel.
        start counting the number of head edges found at 0.

LOOP_1:
        get the image pixel value.

        is the pixel equal to NOTHING.
                no:

                        look around current image pixel, in a 3x3
winodow
                        take values out and test against a head
indicator
                        mask.

                        TestMaskWithWindow( mask1 , 3x3 window).
                        TestMaskWithWindow( mask2 , 3x3 window).
                        TestMaskWithWindow( mask3 , 3x3 window).
                        TestMaskWithWindow( mask4 , 3x3 window).
                        TestMaskWithWindow( mask5 , 3x3 window).
                        TestMaskWithWindow( mask6 , 3x3 window).

                        If any mask results to true.
                                yes:
                                        Set result pixel equal to
WHITE.
                                        Increment head edge
counter.
                                no:
                                        Set result pixel equal to
NOTHING.
```

```
        move to the next pixel on the right.

        are we on the right hand edge of the image.
                yes : move down one pixel, go back to the
                      first lefthand pixel.

        are we on the bottom edge of the image.
                no : goto LOOP_1.

        we have completed the process.
        return the value of the number of head edges found.

//*****************************************
```

CLAIMS

1. A control system for use with a security access device such as a revolving door (1), the control system comprising:

a video input device (6) which in use captures image data from the secured region of the access device;

an image processor (7) for processing the said image data and deriving at least one parameter for discriminating the number of persons present in the secured region of the access device; and

a comparator for comparing the said parameter derived from the original data with a predetermined threshold and for producing a discriminatory output for use in controlling the interlock (3) on the security device depending upon the result of the comparison.

2. A system according to claim 1, in which the image processor (7) and discriminator comprise a cascaded series of modules, each module being arranged to process image data to determine a respective parameter, and to compare the parameter with a corresponding threshold.

3. A system according to claim 2, in which the modules are arranged so that when one module is able to make a decision at a predetermined confidence level then that module produces the said discriminatory output signal, otherwise the said module passing image data onto a subsequent module for further processing.

4. A system according to claim 2 or 3, in which the modules are arranged generally in order of their discriminatory power, with the most powerful module receiving the image data first.

5. A system according to any one of the preceding claims, in which the image processor is arranged to capture a background image of the door with no person present, and to capture a subsequent image of the door with a person present, and to discard from the second said image video data which is unchanged from the background image.

6. A method of controlling a security access device such as a security door including capturing video image data from a secured area of the access door;

processing said image data and thereby deriving at least one parameter for discriminating the number of persons present in the secured area; and

comparing the said parameter with a predetermined threshold and producing a discriminatory output for use in controlling an interlock on the security device to lock the device when more than a predetermined number of persons are present in the secured area.

7. A security access device fitted with a control system according to any of claims 1 to 5.

8. A security access device including a secured region bounded by one or more wholly or partially transparent walls, characterised by a control system including a video input device arranged to view the secured region, and in that the transparent walls include filter means arranged to block or reduce the transmission of light in part of the visible/near-visible optical spectrum, and in that the video input device has a sensitivity/wavelength characteristic generally complementary to the transmission characteristic of the said filter means associated with the transparent walls, the visibility to the video input device of objects outside the transparent walls thereby being reduced or eliminated.

9. A device according to claim 8, in which the filter means are arranged to block transmission in the infra-red, and the video input device is insensitive outside the infra-red.

10. A device according to claim 8 or 9, in which the video input device includes an input optical filter arranged to provide the said sensitivity/wavelength characteristic.

11. A device according to any one of claims 8 to 10 including a control system according to any one of claims 1 to 5.

Fig.1.

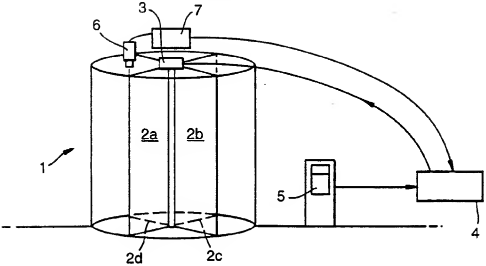
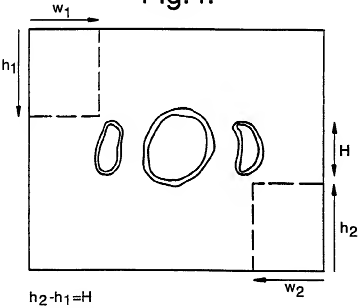


Fig.4.



$h_2 - h_1 = H$
$w_2 - w_1 = W$

PARAMETER H/W

Fig.2A.

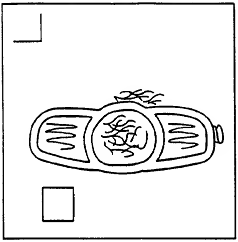


Fig.2B.

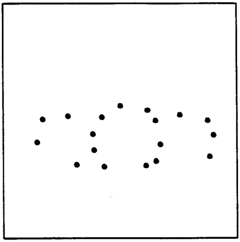


Fig.3.

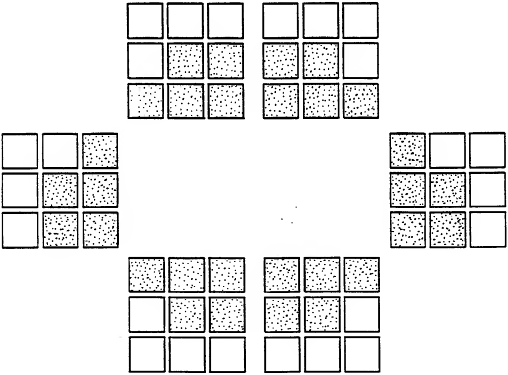
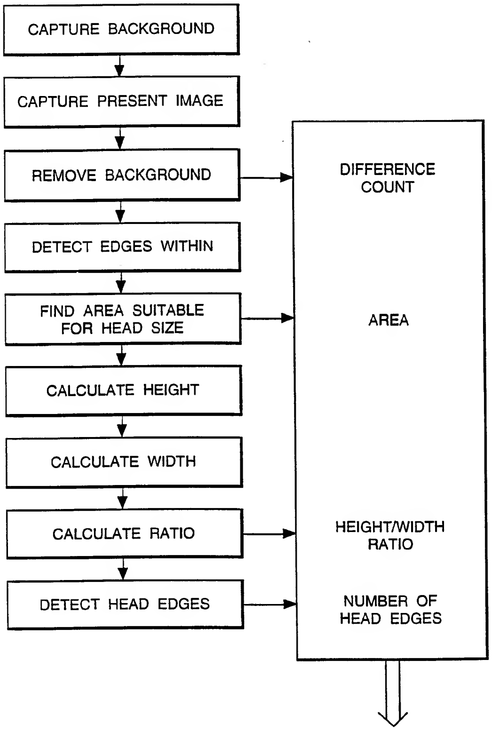


Fig.5.


CAPTURE BACKGROUND
CAPTURE PRESENT IMAGE
REMOVE BACKGROUND
DETECT EDGES WITHIN
FIND AREA SUITABLE FOR HEAD SIZE
CALCULATE HEIGHT
CALCULATE WIDTH
CALCULATE RATIO
DETECT HEAD EDGES
DIFFERENCE COUNT
AREA
HEIGHT/WIDTH RATIO
NUMBER OF HEAD EDGES

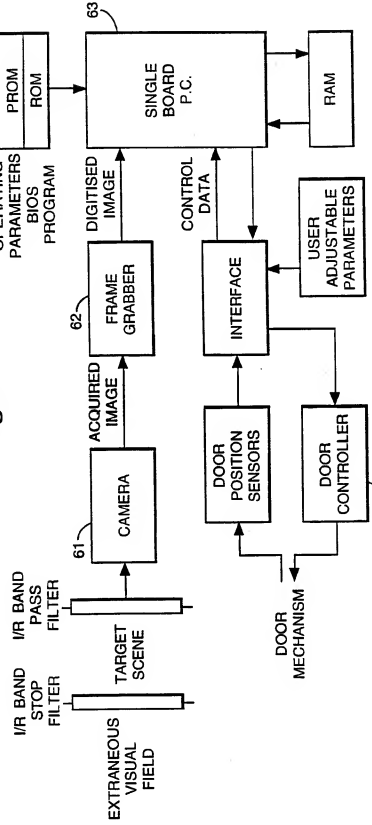

PROM
ROM
OPERATING PARAMETERS BIOS PROGRAM
SINGLE BOARD P.C.
63
RAM
DIGITISED IMAGE
CONTROL DATA
FRAME GRABBER
62
INTERFACE
USER ADJUSTABLE PARAMETERS
ACQUIRED IMAGE
CAMERA
61
DOOR POSITION SENSORS
DOOR CONTROLLER
64
I/R BAND PASS FILTER
TARGET SCENE
I/R BAND STOP FILTER
EXTRANEOUS VISUAL FIELD
DOOR MECHANISM

Fig.6.

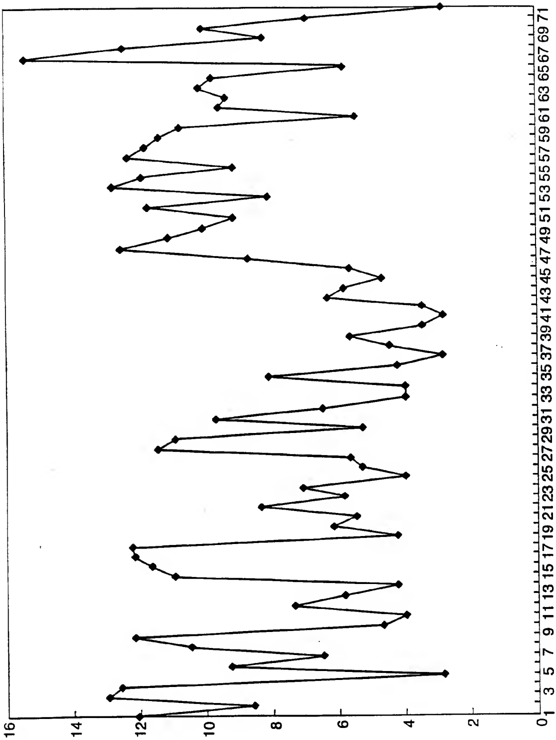

0
2
4
6
8
10
12
14
16
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71

Fig.7a.

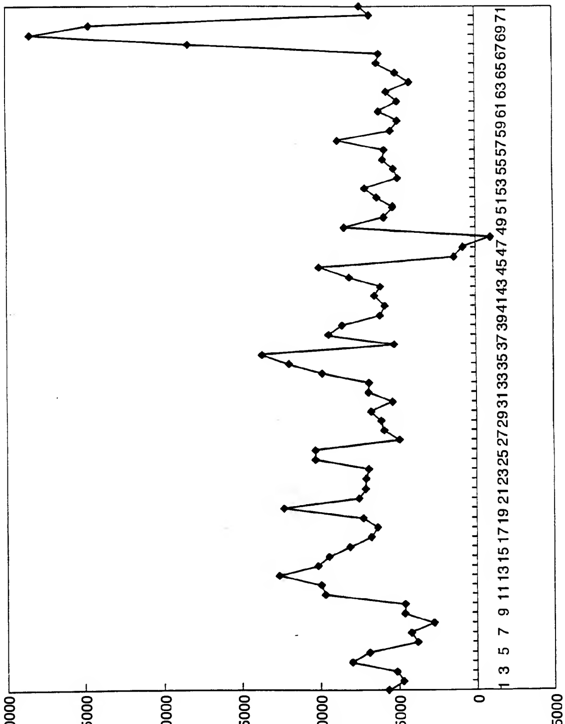

30000
25000
20000
15000
10000
5000
0
-5000
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71

Fig.7b.

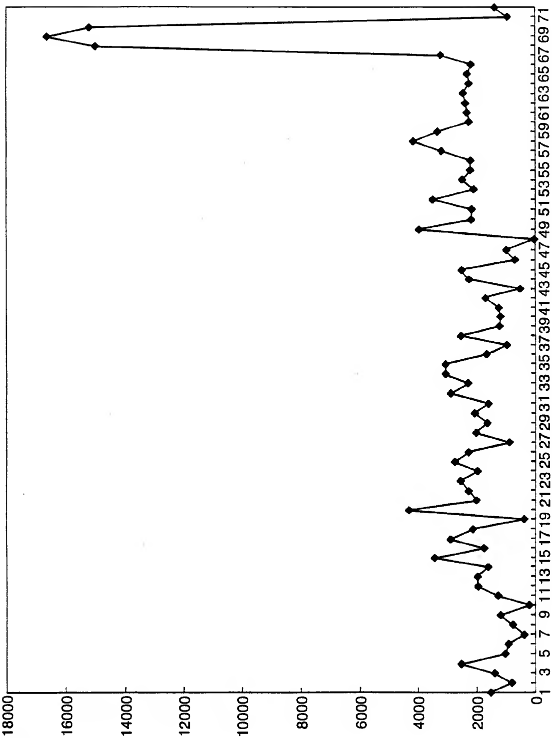
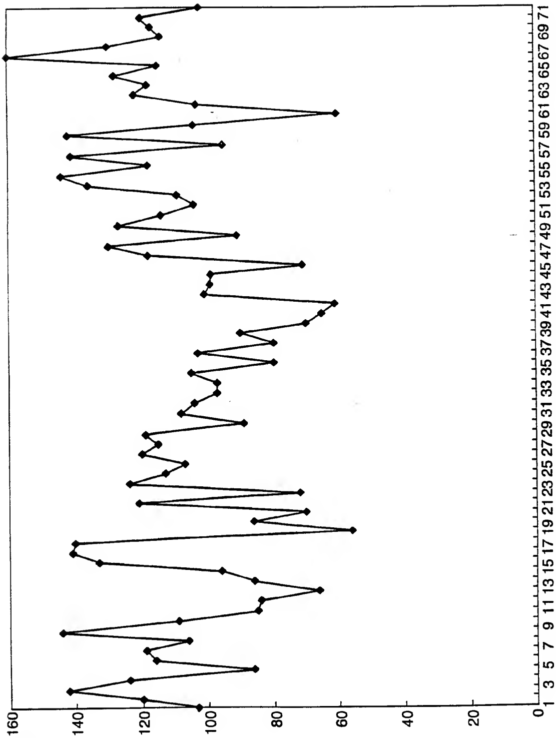

18000
16000
14000
12000
10000
8000
6000
4000
2000
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71

Fig.7c.

Fig.7d.


0
20
40
60
80
100
120
140
160
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71

INTERNATIONAL SEARCH REPORT

Inter   nal Application No
PCT/GB 96/01249

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G07C9/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G07C

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 81 01213 A (OTIS ELEVATOR CO) 30 April 1981 see claim 1; figure 1 ---	1,6-8,11
Y	EP 0 431 363 A (GALLENSCHUETZ E METALLBAU) 12 June 1991 see claim 1; figure 1 ---	1,6-8,11
A	WO 94 27408 A (RCT SYSTEMS INC) 24 November 1994 see claim 1; figure 1 ---	1-11
A	DE 37 40 115 A (MATSUSHITA ELECTRIC WORKS LTD) 9 June 1988 see claim 1; figure 4 --- -/--	1-11

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
11 September 1996	25. 09. 96

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Kirsten, K

International Application No
PCT/GB 96/01249

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 3 564 132 A (BAKER RICHARD H ET AL) 16 February 1971 see claim 1; figure 4 -----	1-11

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/GB 96/01249

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
WO-A-8101213	30-04-81	US-A-	4303851	01-12-81
		CA-A-	1143816	29-03-83
		EP-A-	0037830	21-10-81
EP-A-0431363	12-06-91	DE-C-	3940176	16-05-91
		ES-T-	2047232	16-02-94
		JP-B-	2509385	19-06-96
		JP-A-	4001388	06-01-92
		US-A-	5076013	31-12-91
WO-A-9427408	24-11-94	AU-A-	6786194	12-12-94
		EP-A-	0700623	13-03-96
		US-A-	5465115	07-11-95
DE-A-3740115	09-06-88	JP-B-	7027551	29-03-95
		JP-A-	63266589	02-11-88
		JP-A-	63134981	07-06-88
		JP-B-	6100660	12-12-94
		JP-A-	63134989	07-06-88
		JP-B-	6100661	12-12-94
		JP-A-	63134990	07-06-88
		GB-A,B	2199658	13-07-88
		US-A-	4849737	18-07-89
US-A-3564132	16-02-71	NONE		